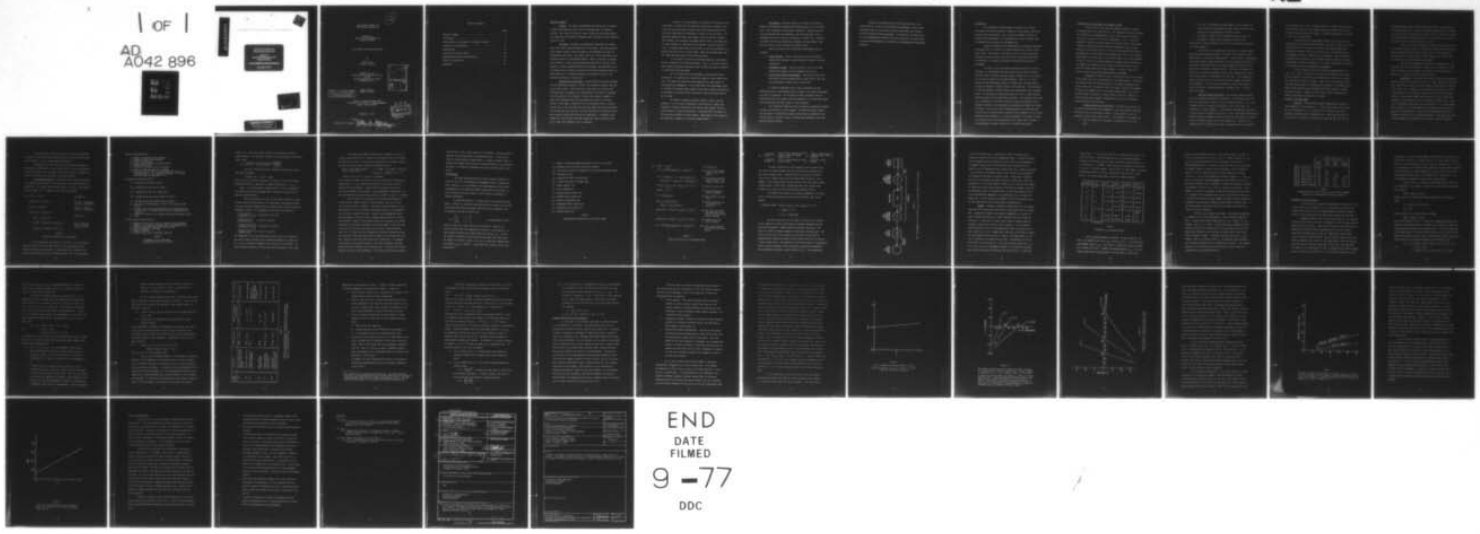
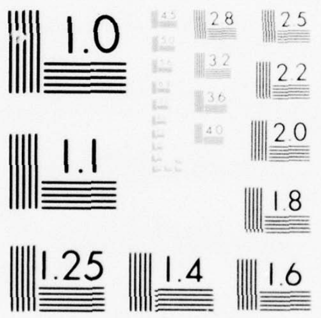
AD-A042 896   ILLINOIS UNIV AT URBANA-CHAMPAIGN CENTER FOR ADVANCED--ETC   F/G 5/2
RESEARCH IN NETWORK DATA MANAGEMENT AND RESOURCE SHARING. A COS--ETC(U)
NOV 75   J D DAY, G G BELFORD   DCA100-75-C-0021

UNCLASSIFIED   UIUC-CAC-DN-75-179   JTSA-5514   NL

1 OF 1
AD
A042 896



END
DATE
FILMED
9 -77
DDC


1.0
4.5
5.0
5.6
6.3
7.1
8.0
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

CAC Document Number 179
JTSA Document Number 5514

Research in
Network Data Management and
Resource Sharing

A Cost Model for Data Distribution

November 1, 1975

3

CAC Document Number 179
JTSA Document Number 5514

Research in Network Data Management and Resource Sharing

A Cost Model for Data Distribution

by

John D. Day
Geneva G. Belford

Prepared for the
Joint Technical Support Activity
of the
Defense Communications Agency
Washington, D.C.

under contract
DCA100-75-C-0021

Center for Advanced Computation
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

November 1, 1975

Approved for release: Peter A. Alsberg
Peter A. Alsberg, Principal Investigator

(See 1473)

Table of Contents

Executive Summary

Purpose. Our goal in undertaking this project was to develop a model describing the various costs of handling data in a network setting. This model could then be used to study the cost effectiveness of various data distribution strategies and to identify the most important sources of cost.

The model. We began by searching the literature for studies which could form a good starting point for our work. The existing model which seemed closest to what we needed is one developed by a group at IBM Research (San Jose). This model [Lum et al., 1975] describes a data staging process in a hierarchical memory. That is, the data is assumed to be stored on a slow, cheap storage device when not in use and transferred to a rapid, expensive device for accessing. What attracted us to this model was its fineness of detail and the ease with which we felt we could extend it to a network situation by including (as part of the hierarchy) devices at a remote site.

Deficiencies in the model. We have identified several problems with the IBM group's approach. First, they implicitly assume a very low rate of data access. Costs which may in fact grow very rapidly with increased load are assumed to be proportional to the number of accesses or to the amount of data handled. Second, they include a number of terms which represent lost CPU time induced by delays in accessing devices. This seems to represent a sort of primitive effort to parcel out the cost of the inevitable CPU idle time among the various processes. At the same time they omit some real CPU costs which are incurred in the data transfer process and which may be significant. In addition, they assume that only CPU idle time adds significant costs and ignore costs due to other idle equipment, such as channels.

In spite of our reservations, we decided to work initally with their model. We felt that the questions we had about it might be more easily and more rationally resolved after we had experimented with it and better understood its limitations, as well as its good points. We therefore began with Lum's cost formula, with its terms for storage, data transfer, and accessing, and added network terms - including costs for data transfer to, from and over the network, as well as protocol costs. In adding these terms, we felt that, if only for consistency, we should follow the spirit of Lum's model. Hence the extended model also has terms involving costs of "lost" CPU idle time.

At this point, we have not introduced important complexities, such as a provision for remote processing. This is a critical omission since remote processing intuitively seems to offer the greatest benefits in distributed data processing.

At its current level of development, the model has severe limitations. The questions we raised with respect to Lum's model carry over. The model only describes data staging and no other aspect of distributed data management. On the other hand, the questionable terms in the model are usually small enough so that their probable inaccuracies are unlikely to seriously affect the kind of broad conclusions that we want to draw.

The model is adequate to make an initial study of the key question: Is it ever more economical to store data at a remote site (instead of locally) and bring it over the net when needed? We have used the model to study this question. We believe that the results of the study have validity for real systems. Improvements in the model are not expected to change our conclusions significantly.

Conclusions. The main result of our study is that heterogeneity is a necessary requirement for remote storage to be cost effective. This conclusion is intuitively reasonable. Transferring data over the network must cost something - and this additional cost is inevitably incurred if the data is stored at a remote site. Therefore the remote site must be significantly cheaper, in some respect, than the local site in order to offset the network costs.

There are several ways in which such heterogeneity may be achieved:

1. Excess capacity. That is, some sites may be less heavily loaded either because of usage patterns or because of system differences.
2. Inexpensive storage. Special facilities, such as the ARPA Network Data Computer, may be available at one site.
3. Artificially-induced heterogeneity. This may be achieved by arbitrarily setting charging rates at some sites so that they are significantly cheaper than at other sites.

It should be emphasized that in most situations the cost differential due to heterogeneity must be sizable - not small percentages, but orders of magnitude. As the amount of data transported over the network decreases, the network costs can decrease to the point where small cost differentials can make remote storage economical.

The interested reader will find an extensive, detailed discussion of these cost balances in this document. He should be warned, however, that the model is sufficiently complex (having some 35 parameters) that careful study is required to gain a thorough understanding of the model and the detailed results.

Finally, we reiterate that the work described here is an initial effort. We have now identified the weaknesses of Lum's model and believe that we can proceed to build a model which more closely describes distributed data management. In particular, we believe that it is a straightforward problem to extend the model to the point where it can be used meaningfully in research on front-ending and intelligent terminals.

Introduction

The advantages of distributing a data base in a network environment have been discussed at length in various papers, panel discussions, and bull sessions. But it has been somewhat difficult to quantify these advantages or to investigate the various tradeoffs and to determine just how great the advantages are.

Several researchers have investigated the problem of optimally allocating files in a network to achieve minimum cost. ([Casey, 1972], [Chu, 1973]). The intent of this paper is to try to gain some understanding of where the major cost factors are incurred and under what circumstances or strategies accessing a distributed file system is worthwhile.

For many of the cost-related questions that arise in the development of a distributed data base system (such as those concerned with the costs of queries, updates, backup, recovery, etc.), the system can at first be viewed as a storage hierarchy. That is, to a local process or user submitting a query to a remote site, storage devices at that site appear as further levels of the hierarchy. From this point of view the network is another channel with some special cost considerations. In this paper we develop and study this sort of simple storage hierarchy model of distributed data processing. This approach will allow us to investigate the tradeoffs offered by various strategies without becoming involved in the complexity of deciding which remote site should be chosen. In fact, what we are attempting here is to determine what criteria such a decision might be based on and the degree of cost control offered by each criterion. In future refinements of the model, we plan to include effects of processing data at the remote sites in order to take advantage of cheaper computation or possible parallelism.

Previous Work on Cost Models for Computer Systems

Cost is both a very vague and ambiguous measure of system performance and a very important one. The ambiguity comes about through the difficulty of assigning dollar costs to all factors of interest. One way, of course, is to carry out experiments - i.e., to run test programs at various sites and compare the bills received. This method yields cost comparisons which are heavily dependent on the pricing policies of the various sites as well as on site hardware and software. Untangling all of these factors to determine what a set of cost figures really means is no easy task. On the other hand, cost is very important in that it serves as an overall measure of system resource utilization. For example, by assigning costs to them, such diverse factors as CPU time and storage used can be added together. In short, costs are a device by which one can add together apples and oranges.

Assignment of specific costs to various factors is of importance to the model user, but not necessarily to the model builder. The latter can consider costs of various resources to be simply weighting coefficients, which can be adjusted at will to reflect a specific environment. It may be, for example, that no real money changes hands. But a user may still wish to evaluate a certain system or piece of software by using a formula which weights storage (which may be in short supply) much more heavily than CPU time.

Modeling network file allocation. Of particular relevance to our study of distributed data management are the cost analyses developed for the network file allocation problem. A good example of such an analysis is that given by Casey [1972]. The parameters in his model are

1. the cost ("mainly for storage") of locating the file at any site,

2. the costs of transmitting a given amount of data between two given sites (with the possibility that update and query transactions may be transmitted at different costs),
3. the amount of update traffic emanating from each site, and
4. the amount of query traffic emanating from each site.

Given values for these parameters, the cost of a particular allocation is readily computed.

Casey states that transmission costs may be "a rather complex monotonically increasing function" of traffic, but he feels that his linear model is a good first approximation. A better idea of transmission costs would require a model which goes into the transmission process in some detail and analyzes the various cost components and how they are affected by the amount of network traffic. The site costs might also profit from a detailed breakdown; note that Casey remarks that factors other than storage are being lumped into one term. It is important to realize, however, that for file allocation Casey's model is probably quite adequate. It is only when one wishes to study other aspects of data distribution - backup and recovery strategies, say - that more detail is needed.

Modeling storage hierarchies. Even before networks existed, the file allocation problem was of importance. The question arose as to where one should place a given file in a storage hierarchy - i.e., a set of memory devices of varying accessibility (core, disk, tape, etc.) connected to a single computer. A particularly comprehensive cost model for this problem has appeared [Lum et al., 1975]. This model differentiates between random and sequential forms of data access and includes considerations of staging, channel costs, CPU overhead, etc. Because of its completeness, we considered this model an appropriate one for extension to the network case. That is, memory devices at a remote site may simply be considered as parts of the storage hierarchy, provided that network costs are properly taken into account. A detailed discussion of the model of Lum et al. appears below.

The distributed data management problem is of course far more complex than the storage hierarchy problem. The model of Lum et al. (and this extension of it) assumes that all data processing (updating and responding to queries) takes place in local core. No provision exists for sending a query to a remote site for processing. Thus, although our straightforward extension of Lum's storage hierarchy model has provided some insight into data distribution, it is grossly inadequate for studying all the many facets of distributed data management.

In what follows we will first review the model described in [Lum et al., 1975]. (In order to facilitate the discussion, this model will be referred to henceforth as the LSWL model.) Next we will extend the LSWL model to include a network. Then we will use the model along with some relevant data to investigate the properties of the model and to analyze some conditions and strategies under which remotely accessing data may be useful. Finally, we will discuss future refinements and further experiments that would be of interest.

A Review of the LSWL Model

Overview. The LSWL model primarily addresses the problem of "data staging" or "data migration". In other words, when a file or data set is not being used (i.e., is inactive) it is stored on one device (usually a relatively slow, inexpensive one). Then, when the data set is accessed, it is moved to a faster, more expensive device so that the program will waste fewer resources waiting for data. The question we are concerned with here is, given the accessing characteristics (number of reads and writes, proportion of time the file is in use, etc.), where in a given hierarchy should the data set be stored when it is inactive and where should it be moved when it is active?

Lum et al. develop an objective function which gives the cost of accessing a data set which is stored on one device when inactive and another (possibly the same device) when active. In this model the entire data set is moved from the inactive device to the active one. (We shall relax this requirement in our model.)

The selection algorithm is then quite straightforward. The objective function is evaluated for a given set of variables for each pair of devices in the hierarchy. The lowest cost then indicates on which pair of devices the data should be located.

Assumptions. The authors make several simplifying assumptions, most of which can be relaxed at the cost of a more complex cost function. They assume that for data sets system paging activity will not significantly affect cost. However, it would probably be necessary to relax this constraint if one wished to consider costs incurred by program activity. They further assume that transfers are direct rather than through core and that there are no flow control problems (i.e., a fast device can always accept data from a slow device). It is also assumed that transfers are not constrained by the capacity of the device the data set is being moved to. These last two assumptions can both be dropped at the cost of a more complex equation. As we shall see, when we add a network to the hierarchy, flow control can not be ignored.

The authors also assume that the data is only staged between two levels, and that multiple staging does not occur (such as disk pack to bulk memory to core, as might happen in Multics).

Although for the most part we carry over these assumptions underlying the LSWL model to our analysis, we will relax the assumption that the entire data set is staged. This will allow us to simulate the ability to retrieve only that part of the data required.

The objective function. Now that we have reviewed the assumptions behind this analysis, let us look at the cost function itself in some detail. The reader should consult table 1 for a key to the symbols used and figure 1 for a summary of the objective function.

$$\begin{aligned}
f_{ij} = {} & \tau_i n_i S' + \tau_j n_j S + && \text{storage cost} \\
& mq[(t_q^{\,j}/\beta) + (s/t_s^{\,j})] + && \text{CPU cost: sequential access + transmission} \\
& mr[t_r^{\,j} + (s/t_s^{\,j})] + && \text{CPU cost: random access + transmission} \\
& \{uq[(t_l^{\,j}/\beta) + (s/t_s^{\,j})] + && \\
& \quad ur[t_l^{\,j} + (s/t_s^{\,j})]\} + && \text{channel cost} \\
& (1 + \lambda)d\{Mw + (S/b_i)[mt_l^{\,i} + (mb_i/t_s^{\,i}) + && \text{cost to move the data set between levels i and j} \\
& \quad (ub_i/t_s^{\,i})] + (mS/B_i)t_c^{\,i}\}\{\Gamma(i - j)\} &&
\end{aligned}$$

Figure 1

Objective Function for the LSWL Model

Let us assume that the data set is at level i of the hierarchy when inactive and at level j when active. (For consistency we will adopt the notation used by Lum et al. whereby the first subscript will be the inactive device, and the second the active one. Also the higher

Data Set Characteristics:

q = number of sequential block assesses.
r = number of random block accesses.
S' = total data set size.
S = amount of data moved to the active level.
s = physical block size.
τ_i = fraction of time data set is on level i.
d = number of times the data set is opened.
λ = the proportion of time to write the data set back to its original position. For read-only data sets, $\lambda = 0$; for full write back at read speed $\lambda = 1$.

Storage Device Characteristics:

t_r^i = random access time for level i.

t_q^i = sequential access time for level i.

t_s^i = transmission rate to or from level i.

t_l^i = average rotational latency time for level i.

t_c^i = minimum access arm movement time for level i.
n_i = unit cost of storage space at level i for the given time period.
b_i = transfer size per access when data set is being moved from a lower level i to another level (or from a higher level to level i).
B_i = largest size that can be transferred without additional access cost.

CPU and Channel Characteristics:

m = adjusted cost per unit time for computer system excluding channel - an estimate of computer wait time induced by I/O
M = unadjusted computer system cost per unit time
u = cost of channel per unit time
β = number of buffers
w = computer setup time for opening a data set

Table 1

Parameters in the LSWL Model
(adapted from [Lum et al., 1975])

levels (i.e., those with faster access) of the hierarchy will have higher indices.) The objective function can be considered to have three major terms:

$$f_{ij} = \begin{Bmatrix} \text{storage} \\ \text{cost} \end{Bmatrix} + \begin{Bmatrix} \text{local process} \\ \text{access costs} \end{Bmatrix} + \begin{Bmatrix} \text{staging} \\ \text{transfer} \\ \text{costs} \end{Bmatrix}$$

The first term is the cost of storing the data on the active and inactive devices.

$$\{\text{storage cost}\} = \tau_i n_i S' + \tau_j n_j S$$

When a data set is moved from level i to level j it is not necessarily deleted from level i; therefore it should be noted that $\tau_i + \tau_j \geq 1$.

(Note: In the LSWL model S always equals S', but to investigate the properties of partial staging and for reasons of clarity we have made this modification.)

The second term is the cost for the user or process to access the data from the active device. This term takes into account the CPU costs and transfer overhead as well as channel costs for both random and sequential accesses. The components of the access cost term are:

$$\begin{Bmatrix} \text{CPU costs for} \\ \text{sequential access} \end{Bmatrix} = mq[(t_q{}^j/\beta) + (s/t_s{}^j)]$$

$$\begin{Bmatrix} \text{CPU costs for} \\ \text{random access} \end{Bmatrix} = mr[t_r{}^j + (s/t_s{}^j)]$$

$$\begin{Bmatrix} \text{channel costs for} \\ \text{sequential access} \end{Bmatrix} = uq[(t_1{}^j/\beta) + (s/t_s{}^j)]$$

$$\begin{Bmatrix} \text{channel costs for} \\ \text{random access} \end{Bmatrix} = ur[t_1{}^j + (s/t_s{}^j)]$$

The components of this term identified as "CPU costs" are measures of the cost of delays incurred by the random and sequential accesses and not of actual resources consumed by the process or in its behalf. For a more lengthy discussion of these costs and the quantity m, see below.

The final term (staging transfer costs) computes the cost of moving the data from level i to level j and includes factors for writing the data back to level i if necessary, preparation for transfer, latency waiting for the next block, and block transmission costs.

$$\left\{\begin{matrix}\text{cost to move data between} \\ \text{level i and level j}\end{matrix}\right\} = (1 + \lambda)d\{Mw + (S/b_i)[mt_1^{\,i} + (mb_i/t_s^{\,i}) + (ub_i/t_s^{\,i})] + (mS/B_i)t_c^{\,i}\}\Gamma(i - j),$$

where $\Gamma(x)$ is 0 if $x = 0$ and is 1 otherwise.

Notice that this model says that if, say, only 10 percent of the data is shipped back ($\lambda = 0.1$), then only 10 percent of the setup cost Mw is incurred by this operation. Clearly this is incorrect; the cost of setup is independent of the amount of data subsequently transferred. We have therefore corrected the setup term in our model to read $(1 + \Gamma(\lambda))Mwd$.

At this point it is appropriate to discuss the parameter m in some detail. When a process or user accesses a data set, it must wait for this access to complete. This delay consists primarily of the time required to set up the device (rotational latency or arm movement) and the time to transfer the data. Clearly, multiprogramming systems take advantage of this wait time by allowing other processes to utilize the processor. However, these delays, which are incurred by all running processes in the system, contribute to the total amount of CPU idle time. To account for this lost time Lum et al. define an "adjusted machine cost", m. For lack of a better formulation, they have defined this cost to be percent of CPU idle time times the dollar cost associated with the CPU. There are some difficulties with such a definition. For example, as the load on the system increases, so may CPU utilization, queueing delays and system overhead, thus increasing cost. The objective function does not account for this phenomenon. This characterization also assumes that the CPU is the crucial resource to be utilized. Current trends in hardware could actually make this assumption false. It may also be false for certain specific applications. It might be equally valid to include idle channel time incurred by a process because it was using the processor. We intend to investigate this issue in more detail in the future.

Network Model

The model discussed here will require further extensions to model the cost of a distributed data management system in complete detail. However, it is a reasonable first approximation and will allow investigation of the tradeoffs between storage and access economy, as well as provide an accurate model of file or data set staging in a network environment.

As mentioned earlier, a primary concern in extending the LSWL model to allow for a network in the hierarchy is to account for the flow control and other protocol-related costs that will be incurred. The cost function used has the basic form:

$$c_{ij} = \left\{ \begin{array}{ll} f_{ij} & i > k \\ g_{ij} & i \leq k \end{array} \right\} \qquad \text{(j always greater than k)}$$

where k is the first remote level of the hierarchy. (Here we are tacitly assuming that all staging will be done to a local device.) We have already discussed the original objective function, f_{ij}. We will now proceed to consider the cost function that deals with the network. The reader is directed to table 2 for a key to additional symbols and to the summary of g_{ij} in figure 2. The network cost function can be characterized as:

e	= number of message exchanges necessary to set up the transfer
t_{nd}	= message round trip delay time in the network
t_{np}	= CPU time for protocol overhead (on a per protocol message basis)
K	= compression factor
t_{nr}	= network CPU time to receive data
t_{nt}	= network CPU time to transmit data
u_r	= remote channel cost
u_L	= local channel cost
m_r	= adjusted remote system cost
m_L	= adjusted local system cost
n_k	= network transmission cost
M_r	= unadjusted remote system cost
M_L	= unadjusted local system cost
b_k	= network packet size

Table 2

Supplementary Parameter List for Network Model

$g_{ij} = \tau_i n_i S' + \tau_j n_j S +$	(1)	storage cost
$(1 + \lambda)d\{(SK/b_k)[m_r b_k/t_s^k + u_r b_k/t_s^k]\} +$	(2)	cost to move between highest remote level and net
$\{(1 + \Gamma(\lambda))dM_r w\} + (1 + \lambda)d\{(S/b_i)[m_r t_1^i + (m_r b_i/t_s^i) + (u_r b_i/t_s^i)] + (m_r S/B_i)t_c^i\} +$	(3)	cost to move between inactive level i and highest remote level
$de\{(m_r + m_L)t_{nd} + (M_r + M_L)t_{np}\}\{1 + \Gamma(\lambda)\} +$	(4)	protocol setup cost
$2en_k d\{1 + \Gamma(\lambda)\} +$	(5)	network charges for protocol messages
$(1 + \lambda)(SKn_k/b_k)d +$	(6)	data transfer network costs
$(M_r t_{nt} + M_L t_{nr})(S/b_k)d + \lambda(M_r t_{nr} + M_L t_{nt})(S/b_k)d +$	(7)	network software cost to send data and receive it
$m_L q[(t_q^j/\beta) + (s/t_s^j)] + m_L r[t_r^j + s/t_s^j] +$	(8)	CPU costs for random and sequential access and for retrieval from active location
$u_L q[(t_1^j/\beta) + (s/t_s^j)] + u_L r[t_1^j + (s/t_s^j) +$	(9)	channel costs for local retrieval
$(1 + \lambda)d\{(SK/b_k)[(m_L b_k/t_s^k) + (u_L b_k/t_s^k)]\}$	(10)	cost to move between net buffers and active device

Figure 2

Objective Function for the Network Model

$$g_{ij} = \left\{\begin{matrix}\text{storage}\\ \text{cost}\end{matrix}\right\} + \left\{\begin{matrix}\text{cost to move between inactive}\\ \text{remote level and highest}\\ \text{remote level}\end{matrix}\right\} + \left\{\begin{matrix}\text{cost to move between}\\ \text{highest remote level}\\ \text{and net}\end{matrix}\right\}$$

$$+ \left\{\begin{matrix}\text{network}\\ \text{costs}\end{matrix}\right\} + \left\{\begin{matrix}\text{cost to move between net and}\\ \text{active level}\end{matrix}\right\} + \left\{\begin{matrix}\text{process access}\\ \text{costs}\end{matrix}\right\}$$

The major differences in this equation from the purely local version are the added network costs and the distinction between local and remote charging rates. Otherwise most of the terms are special cases of the original and we will not discuss them in detail. For a summary of the staging process and the various costs, the reader should consult figure 3, which shows schematically where the various terms (labeled as in figure 2) enter into the data transfer process.

The network costs consist of two major components: the setup costs for using the network and the cost of the traffic sent on the network.

$$\left\{\text{network costs}\right\} = de\{(m_r + m_L)t_{nd} + (M_r + M_L)t_{np}\}\{1 + \Gamma(\lambda)\}$$
$$+ \; 2en_k d\{1 + \Gamma(\lambda)\}$$
$$+ \; (1 + \lambda)\;(SKn_k/b_k)d$$

The first term (term (4) in figure 2) is the cost of setting up the transfers in terms of the number of message exchanges required (protocol negotiation), network delay and protocol processing. The other two terms are network charges for the packets actually sent. The first of these (term (5) in figure 2) is the cost for the protocol negotiation and connection setups, and the second (term (6)) is the cost of data actually sent. The constant K in this last term is a "compression" factor to allow inclusion of data compression and protocol overhead in data transmission (headers, restart markers, etc.). The transmission

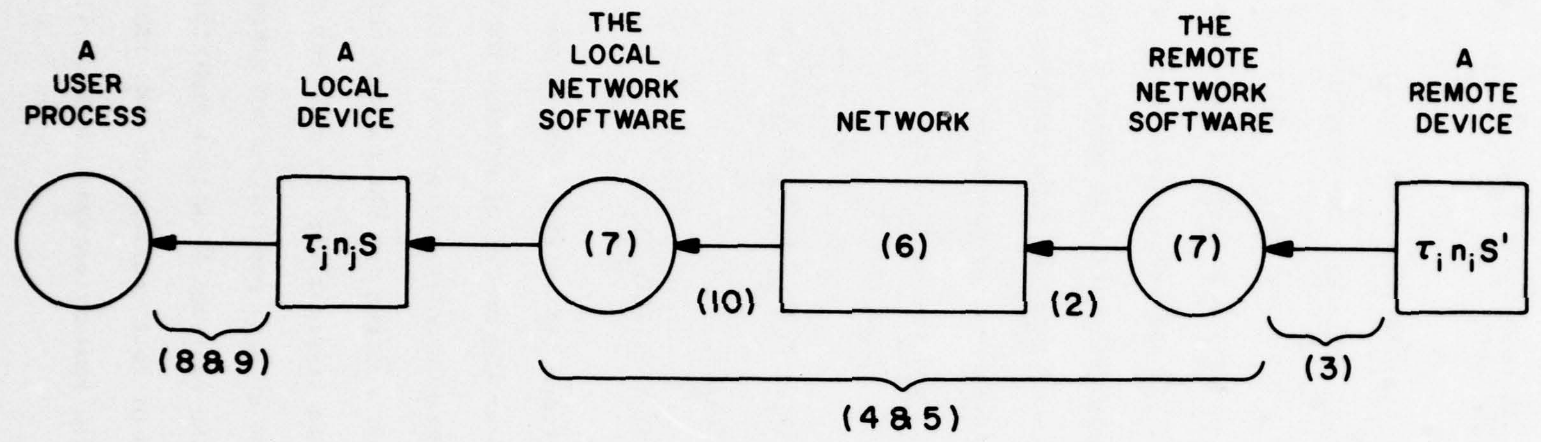


Figure 3

The correspondence of the terms of g_{ij} to the major aspects of the system.

cost of the network, n_k, is calculated in terms of packets sent, a charging structure in use in the commercial world. (It should be noted that the symbols with the subscript k do not refer to the properties of the highest remote level of the hierarchy but to properties of the network, such as transmission rate, packet size, etc.) Factors involving λ are included in the network costs to take account of the possibility of shipping the data back to inactive store. Notice that a transfer must be set up no matter how small an amount is sent back - hence the appearance of $\Gamma(\lambda)$ in the formula. Terms (2), (7), and (10) (see figure 2), which are costs of data transfer to and from the network, will also be considered as part of "network costs" in our later analysis, since they form important components of the additional cost of storing at a remote site. But in form they are similar to the local transfer costs of Lum's model and so do not need further discussion here.

Example. Consider a situation in which there is a four-level hierarchy (core, drum, disk, and archive), both locally and at a remote site. Assume that values of the relevant parameters are as given in table 3 (taken from Lum et al. [1975]) and that they are the same at both sites. It does not, of course, make sense to consider inactive storage at remote core, and this case is omitted. Let the number of local buffers be two ($\beta = 2$) and assume that there is no setup time to open a data set ($w = 0$). Suppose that a data set of 10^8 bytes is active for one eight-hour shift per day, so that on a per-month basis $d = 30$ (i.e., the data set is opened once per day). Furthermore, the set is then active 1/3 of the time ($\tau_j = 1/3$), and we shall assume that $\tau_i = 1$ (i.e., that the set is permanently resident at the inactive location). Let the set be blocked into 1500-byte physical records ($s = 1500$) and suppose that $\lambda = 1$ (so that the data set is always written back at the end of each day). Finally assume that there are 90,000 sequential accesses to the active copy per month and 210,000 random accesses (i.e., $q = 90,000$ and $r = 210,000$). These values all correspond to those used by Lum et al. in their example. Notice that the total number of accesses (300,000 per month) is very low, amounting to less than one I/O per second. The reader should keep in mind this hidden assumption.

Parameter	Core	Drum	Disk	Archive	Unit
t_r^i	10^{-6}	5×10^{-3}	60×10^{-3}	5	second
t_s^i	∞	10^6	3×10^5	5×10^4	byte/sec
t_q^i	0	8×10^{-3}	13×10^{-3}	25×10^{-3}	second
t_l^i	0	8×10^{-3}	12×10^{-3}	20×10^{-3}	second
t_c^i	0	0	25×10^{-3}	40×10^{-3}	second
n_i	2×10^{-2}	5×10^{-4}	3×10^{-5}	3×10^{-7}	\$/byte/month
b_i	*	20,000	7,000	2,000	byte
B_i	*	4×10^6	140,000	10,000	byte

* Irrelevant

Table 3

Parameters for Storage Hierarchy

Next, network parameters are needed. We have taken $b_k = 125$ bytes, the ARPANET packet size; $t_{nd} = 200$ ms and $t_s^k = 5 \times 10^3$ bytes/sec, both ARPANET figures; $t_{np} = 1$ ms, which is roughly the time for an ARPA NCP to handle one protocol command (including response); $t_{nr} = 1$ ms, an average figure which runs from about .5 ms NCP time to 2 ms if the process must be awakened; and t_{nt} = 2 ms, which consists of about 1 ms to get to the NCP and 0.5 to 1 ms to use it. (These estimates for t_{np}, t_{nr}, and t_{nt} were supplied to us by G. Grossman of the Center for Advanced Computation.) It should be noted that both t_{nr} and t_{nt} should be slightly larger to allow for data processing by the file transfer protocol. This is particularly true if data compression is being carried out. But for this example we initially assume $K = 1$. Also, t_{nr} and t_{nt} as given are times per message; we have divided by 8 to get a per-packet estimate, since a maximum of 8 packets per message is allowed. The parameter e was set at 15. This is arrived at as follows. In the ARPANET, it requires 7 exchanges to open an FTP connection, plus from 4 to 7 commands to set parameters and 3 more to open the data connection. It should be noted that by using ARPANET data and the values supplied by Grossman we are essentially computing lower bounds on network costs. In other environments the network costs will be higher and results are likely to be quite different.

Finally, cost estimates are needed. For network transmission we assumed n_k = \$1.25 per 1000 packets, a quoted Telenet commercial rate. To begin with we have assumed that $m_L = m_r$ = \$10/hr., $M_L = M_r$ = \$100/hr., and $u_L = u_r$ = \$8/hr. Clearly under these assumptions remote storage will not be cost effective; but by adjusting the cost of the remote site relative to that locally, we should reach a point where remote storage is cheaper. The values calculated for costs c_{ij} (see figures 1 and 2) are given in table 4. As expected, remote storage is far from being economical for the assumed cost structure. The cheapest method is for the inactive data to be stored on local archive and transferred to local disk when active.

		Active Location (j)			
		Local Core	Local Drum	Local Disk	Local Archive
Inactive Location (i)	Local Core	2000			
	Local Drum	717	50.0		
	Local Disk	670	19.8	3.05	
	Local Archive	668	17.5	1.91	3.01
	Remote Drum	789	139.0	123.0	125.0
	Remote Disk	742	92.3	76.7	78.6
	Remote Archive	740	90.0	74.4	76.4

Table 4

Computed values of total costs c_{ij} for the basic example.
Entries are in thousands of dollars per month.

Analysis of the Cost Formula

In this section we attempt an assessment of the effects of the various terms in the formula for g_{ij}. In particular, we look at the formula from the point of view of determining what range of parameter values or cost differentials will make remote storage cost effective.

Comparing figures 1 and 2, notice that terms (8), (9), and the second part of term (1) (the cost of storage on the local staging device) appear in both f_{ij} and g_{ij}. They involve only local costs and belong to what might be called the post-staging phase of the access process. These terms therefore play no role in a comparison of the absolute costs of local and remote storage. They do, however, play a role in the study of relative costs, since, if the staged data is used very heavily, terms (8) and (9) may form a large part of the total. The same holds for term (1), if a large amount of data is staged and the staging storage device is costly, as it usually is. In this section, however, we shall consider $g_{ij} - f_{ij}$ and so shall ignore terms (8) and (9), as well as the second part of term (1). We shall also set λ to zero, since a non-zero value can at most introduce a factor of two into transfer costs. (It is also reasonable to anticipate building some more rational update mechanism into the model than a simple shipping of a large fraction of the data (presumably modified) back to the original site.) We also set $\tau_i = 1$, corresponding to permanent storage on the inactive device.

Term (3) in g_{ij} - a transfer cost between levels - also has its counterpart in f_{ij}; namely, the last term. Term (3) may be written more simply as

$$d[M_r w + S(m_r \phi_i + u_r \Psi_i)],$$

where ϕ_i and Ψ_i are functions involving properties of the inactive device:

$$\phi_i = t_1^{\ i}/b_i + 1/t_s^{\ i} + t_c^{\ i}/B_i$$

$$\Psi_i = 1/t_s^{\ i}$$

The last term in f_{ij} is quite similar, reading

$$d[M_L w + S(m_L \phi_i + u_L \Psi_i)].$$

Here we have omitted the $\Gamma(i - j)$ factor for comparison purposes; this omission is justifiable since it is rarely cost effective to make the staging storage the same as the inactive store. If we also assume that the cheapest device for inactive store (either local or remote) is the same at both sites (so that ϕ_i and Ψ_i are the same in both f_{ij} and g_{ij}), then we obtain the following expression for differential cost:

$$\begin{aligned} g_{ij} - f_{ij} &= d[(M_r w_r - M_L w_L + S\phi_i(m_r - m_L) + S\Psi_i(u_r - u_L)] \\ \text{(A)} \qquad &\quad + S'(n_{ir} - n_{iL}) \\ &\quad + \{\text{Terms } (2) + (4) + (5) + (6) + (7) + (10)\}. \end{aligned}$$

Here we have used n_{ir} and n_{iL} to distinguish between the remote and local costs for inactive storage, and w_r, w_L to indicate remote and local file setup times, respectively.

We wish to investigate under what conditions $g_{ij} - f_{ij}$ is approximately zero. Consider first what happens when we neglect the network protocol costs (terms (4) and (5)), and also the setup time (i.e., we set $w_r = w_L = 0$, as does Lum). (The conditions under which terms (4) and (5) are relatively small are discussed below. Setting $w = 0$ is invalid for many systems; the consequences of a non-zero w will also be discussed further below.) The expression for $g_{ij} - f_{ij}$ now looks like:

$$\begin{aligned} g_{ij} - f_{ij} &\simeq dS[(m_r - m_L)\phi_i + (u_r - u_L)\Psi_i] \\ &+ S'[n_{ir} - n_{iL}] \\ &+ \{\text{Terms } (2) + (6) + (7) + (10)\}. \end{aligned} \tag{B}$$

It is important to notice that the four bracketed terms contain a common factor of Sd. Hence we can make the following immediate remarks about the approximate expression (B).

1. If $S = S'$, or if $n_{ir} = n_{iL}$, the parameter S (the amount of data transferred) has no effect on which storage (local or remote) is cheaper. The cost differential is, of course, proportional to S; however, relative costs are independent of S.

2. If remote and local storage costs are equal ($n_{ir} = n_{iL}$), the expression given in (B) has a common factor d (the number of times the data transfer takes place). Thus the role played by d in the cost comparison is similar to that played by S, as discussed in the preceding remark. Equality of storage costs is probably a very realistic approximation. Since the cheapest

inactive storage devices at the two sites are likely to be identical (or very similar), there is a valid basis for assuming a negligible cost differential.

The two preceding remarks merely serve to indicate some factors which do <u>not</u> help to make remote storage cost effective. There are only two features of our model which <u>can</u> help to make remote storage cost effective. These are

a) a lower cost for term (3) than occurs for the comparable term in f_{ij}, and

b) a lower cost for remote inactive storage than for local inactive storage.

To get some idea of how great the savings must be, we note that even if local costs are large and remote costs are zero, the network costs (including cost of transfer to and from the net) may be large enough so that remote storage is not economical. Specifically, this will occur when (from (A))

(C) Terms (2) + (4) + (5) + (6) + (7) + (10)

$$> d[M_L w_L + S\phi_i m_L + S\Psi_i u_L] + S' n_{iL},$$

where $m_r = M_r = 0$ in the network terms.

In view of the preceding comment, it is worthwhile to tabulate estimates of the magnitudes of the network terms for closer analysis. Table 5 contains a listing of the network terms in a format convenient for comparison and estimation. In each term, factors independent of the storage and transfer strategies or of host charging policies have been lumped into a single parameter and a careful estimate of this parameter has been made. In cases where the parameter may vary widely, bounds are given. If the variation is not likely to be as much as an order of

Term Number	Description	Simplified Form	Bounds or Estimates of Constant	Units of Constant
(2)	cost to move between highest remote level and net	$dSK(m_r + u_r)C_2$	$3 \times 10^{-10} \leq C_2 \leq 5 \times 10^{-7}$	hours per byte
(4)	protocol setup cost	$de(M_r + M_L)C_4$	$C_4 \simeq 5 \times 10^{-6}$	hours per message
(5)	network charges for protocol messages	deC_5	$C_5 \simeq 2.5 \times 10^{-3}$	dollars per protocol interchange (message and reply)
(6)	data transfer network costs	$dSKC_6$	$C_6 \simeq 10^{-5}$	dollars per byte
(7)	network software cost to send data and receive it	$dS(M_r + M_L)C_7$	$10^{-9} \leq C_7 \leq 10^{-8}$	hours per byte
(10)	cost to move between net buffers and active device	$dSK(m_L + u_L)C_2$	(see above)	hours per byte

Table 5

Network terms simplified and tabulated for easy comparison. $\lambda = 0$ is assumed throughout. In Term (4) it is also assumed that $m_r = 0.1M_r$ and $m_L = 0.1M_L$.

magnitude, an average value is given. A number of further remarks may be derived immediately from inspection of table 5. These are:

3. Term (6) (data transfer cost) is generally the largest of the network terms by about an order of magnitude.
4. Terms (2) and (10) (cost of transfers between net and host) become comparable to Term (6) only when the constant C_2 is at or near its upper bound. This situation corresponds to very small network bandwidth (t_s^k about 500 bytes per second).
5. Term (7) (network software cost of data transfer) is small compared to Term (6) unless one of the following conditions holds:
   a) CPU time is very expensive,
   b) network software is more inefficient than assumed, or
   c) the compression factor K is unrealistically small.
6. The protocol costs (Terms (4) and (5)) are about equal to each other, although Term (5) dominates if CPU time is relatively cheap. Both of these terms tend to be negligible compared to Term (6). That is, they are an order of magnitude smaller unless the amount of data transferred (SK) is very small (less than about 5×10^4 bytes).
7. In summary, for most situations the totality of the network terms may be approximated by Term (6) and hence estimated to be $dSK \times 10^{-5}$.*

* The constant here has dimensions dollars/byte. The reader should be warned that by consolidating constants in this analysis we have sometimes generated expressions which may appear dimensionally bizarre. But the units which must apply to the constants are readily reconstructed.

Returning to comparison (C) above, we see now that, to a good approximation, remote storage cannot be cheaper than local storage as long as

(C') $\quad dSK \times 10^{-5} > d[M_L w_L + S\phi_i m_L + S\Psi_i u_L] + S' n_{iL}.$

(That is, the remote strategy is then more expensive no matter how cheap remote storage and processing costs are.) Notice that if, as was assumed earlier, $w_L = 0$ and $S = S'$, then inequality (C') simplifies to

(C'') $\quad K \times 10^{-5} > \phi_i m_L + \Psi_i u_L + n_{iL}/d.$

The right side of this inequality must be investigated further. Using the parameter values given in the example of the preceding section, we find that for archival storage $\phi_i \simeq 10^{-8}$ and $\Psi_i \simeq 5 \times 10^{-9}$ (both in units of hours per byte). For disk or drum these factors are considerably smaller. Hence the numbers given are rough upper bounds on ϕ_i and Ψ_i. We immediately conclude that $\phi_i m_L$ and $\Psi_i u_L$ are smaller by orders of magnitude than the left side of (C'') and hence cannot contribute to making remote storage cost effective. Furthermore, for archival storage we assumed $n_i = 3 \times 10^{-7}$; hence the term n_{iL}/d is negligible also. We therefore can add to our list of remarks:

8. If w_L is small and $S = S'$, network costs far outweigh any potential savings from the remote site's being cheaper (or free).

9. If w_L is small, but $S \neq S'$, free remote storage becomes cost effective when

$$S'/S > \frac{dK \times 10^{-5}}{n_{iL}}$$ (or when less than about 0.1 percent of the data base is staged). If remote storage is not free, it still may become cost effective, specifically when

$$S'/S > \frac{dK \times 10^{-5}}{n_{iL} - n_{ir}}.$$

10. If w_L is large but w_r is negligible (so that $M_r w_r$ is negligible as is assumed in (C')), then the large local setup cost $M_L w_L$ may make remote storage economical (even if $n_{ir} \simeq n_{iL}$). To be specific, suppose $w_L = 1$ sec. (Setup times of this magnitude occur in some operating systems.) Then (from (C')) remote storage will be cost effective due to the setup time differential whenever

$$SK < M_L w_L \times 10^5 = 30\ M_L,$$

or, for, say, $M_L = \$200$ per hr., $SK < 6 \times 10^3$.

<u>Computational Results and Conclusions</u>

At this point it is probably a good idea to remind the reader of the limitations of this model. The model depicts the cost of a program which accesses data that reside at some remote site. No attempt is made to consider the advantages of remote processing, of multiple copies for reliability, etc., although some indirect implications along these lines are possible. We can, however, use the model to investigate various strategies (such as local caching of data) and to evaluate their effectiveness in utilizing remote resources under various conditions. This section contains the results of such experiments. The graphs of this section have all been generated using the basic parameter values listed for the example discussed in detail earlier; that is, all parameter values not specified in text or figure caption are to be assumed those given in the example. Thus results are to be interpreted as holding in the general context of that basic example. As we discussed in the section just preceding, some system parameters are subject to wide variation, and changing them can have a dramatic effect on relative sizes of terms, as well as on absolute total costs.

The basic result one finds from manipulating this model is that the network must be heterogeneous in order for remote storage to provide any cost advantage. There are several ways in which this heterogeneity may be achieved:

1. Excess capacity. This may be achieved either by having a network of similar systems in which one or more are not heavily loaded or by having different systems that can take advantage of their differences (speed, special hardware, etc.) to generate excess capacity.
2. Inexpensive storage. This may be achieved by either charging policies or by special facilities such as the ARPA Network Data Computer, laser stores, etc.
3. Artificially-induced heterogeneity. This may be achieved by politically setting charging rates at some sites so that they are significantly cheaper than at other sites. This last method can be fairly dangerous to implement as can happen when reality is traded for illusion. Experience has shown that, if charges are sufficiently low (or free), management, as opposed to users, will tolerate incredibly poor response in order to use only that resource.

Let us first consider what effect attempts to introduce heterogeneity into system cost have on overall cost. To introduce heterogeneity we will set M_r, m_r, and u_r to be some fraction, Z, of M_L, m_L and u_L, respectively. This differential can be considered to be caused by different hardware, different system loads, or different charging policies on the local and remote systems. For this situation as in all others discussed in this section we are only considering cases in which a part of the data set is moved (i.e., $S \neq S'$). This situation is intended to correspond to the process' only moving the data it needs. As one can see by looking at figure 4, Z has little effect on the overall cost for a data staging model. This result is not surprising if we consider that, for these values of S and S', about 75% of the cost ($46) is in storage charges (Term 1) and network packet charges (Term 6). In addition, about $9 is spent on local accessing (Terms (8) and (9)). Therefore Z may have a larger effect in remote processing environments in which storage and net charges would not constitute such a large fraction of the cost (i.e., if relatively more processing time is consumed in the staging). However, lowering remote storage charges and compressing the data for shipment over the network can produce a remote strategy which provides significant savings over the local strategy, as can be seen from figures 5 and 6. The availability of exotic mass stores (such as the laser memory, which can provide one or even two orders of magnitude differential in price) can make a remote strategy a very viable one. Notice that the results pictured in figures 5 and 6 may be compared with Remark 9 in the analysis of the preceding section. From that remark, the crossover point (where remote storage becomes cost effective) can be estimated to be $S = 5 \times 10^4$ for $n_{ir} - n_{iL} = 1.5 \times 10^{-7}$ and $K = 1$. Figure 5 shows this crossover at $S = 1.7 \times 10^4$. This agreement is quite reasonable; much of the discrepancy can be attributed to the assumption in the analytical study that $\lambda = 0$. A similar comparison holds for the other crossovers shown.

It is interesting to note that protocol costs and network-related host software costs make up a fairly small fraction (normally less than 10%) of the total cost of data staging. (See the section

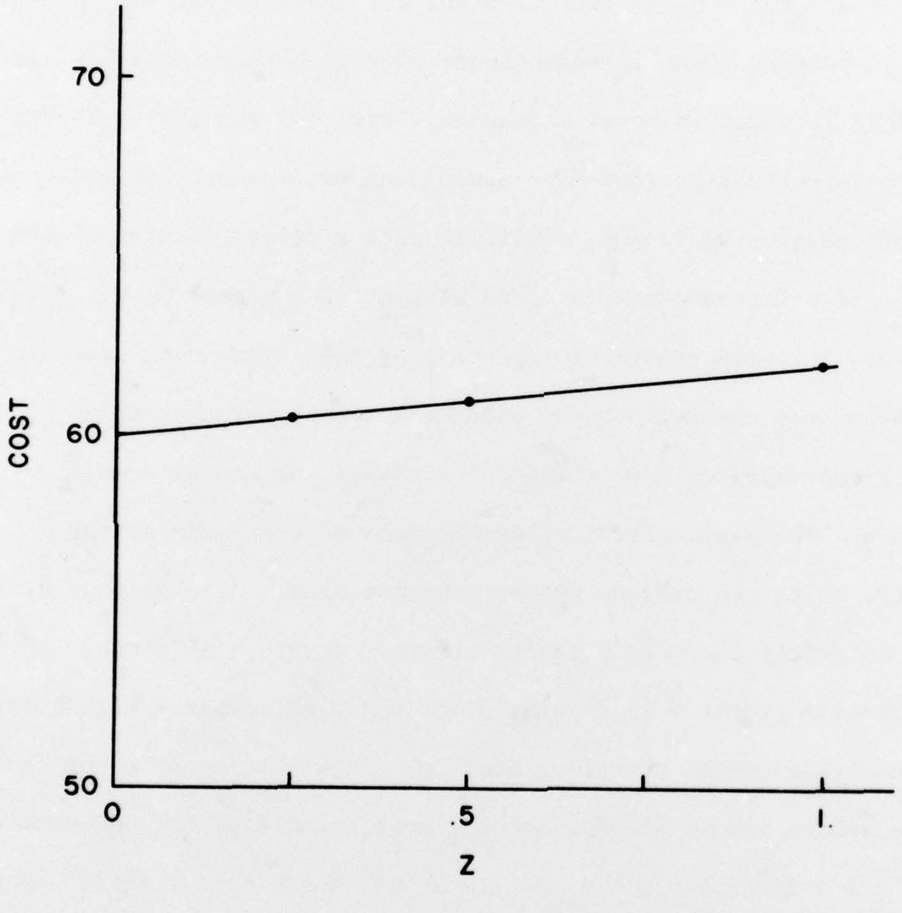


Figure 4

Cost of staging from remote archive to local drum as a function of Z. $S' = 10^8$; $S = 2 \times 10^4$. Costs are in dollars per month.

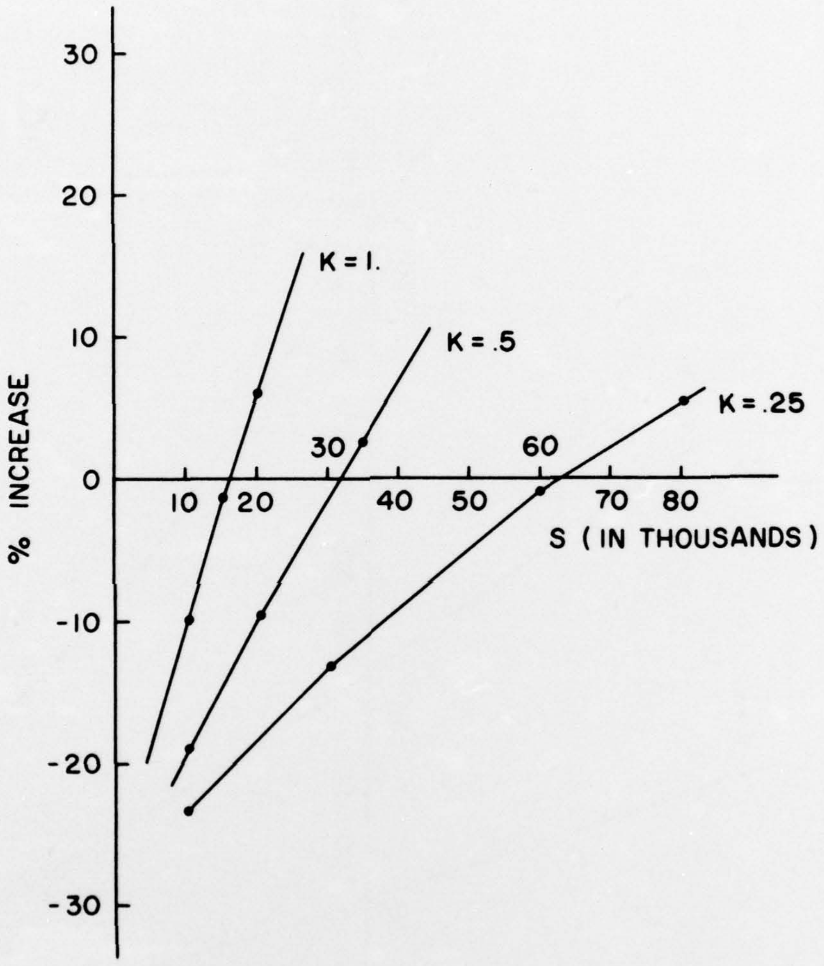


Figure 5

The percent increase in cost of remote over local strategy as a function of the amount of data moved. Percent increase is computed as $(g_{ij} - f_{ij}) \times 100/f_{ij}$, where f_{ij} is the best strategy for local inactive store and g_{ij} is best for remote inactive store. Remote storage devices are assumed to cost half as much as local ones; local costs are as given in table 3. The effect of varying the compression factor K is also shown. For convenience, computed points are joined by straight lines; the curves are actually smooth.

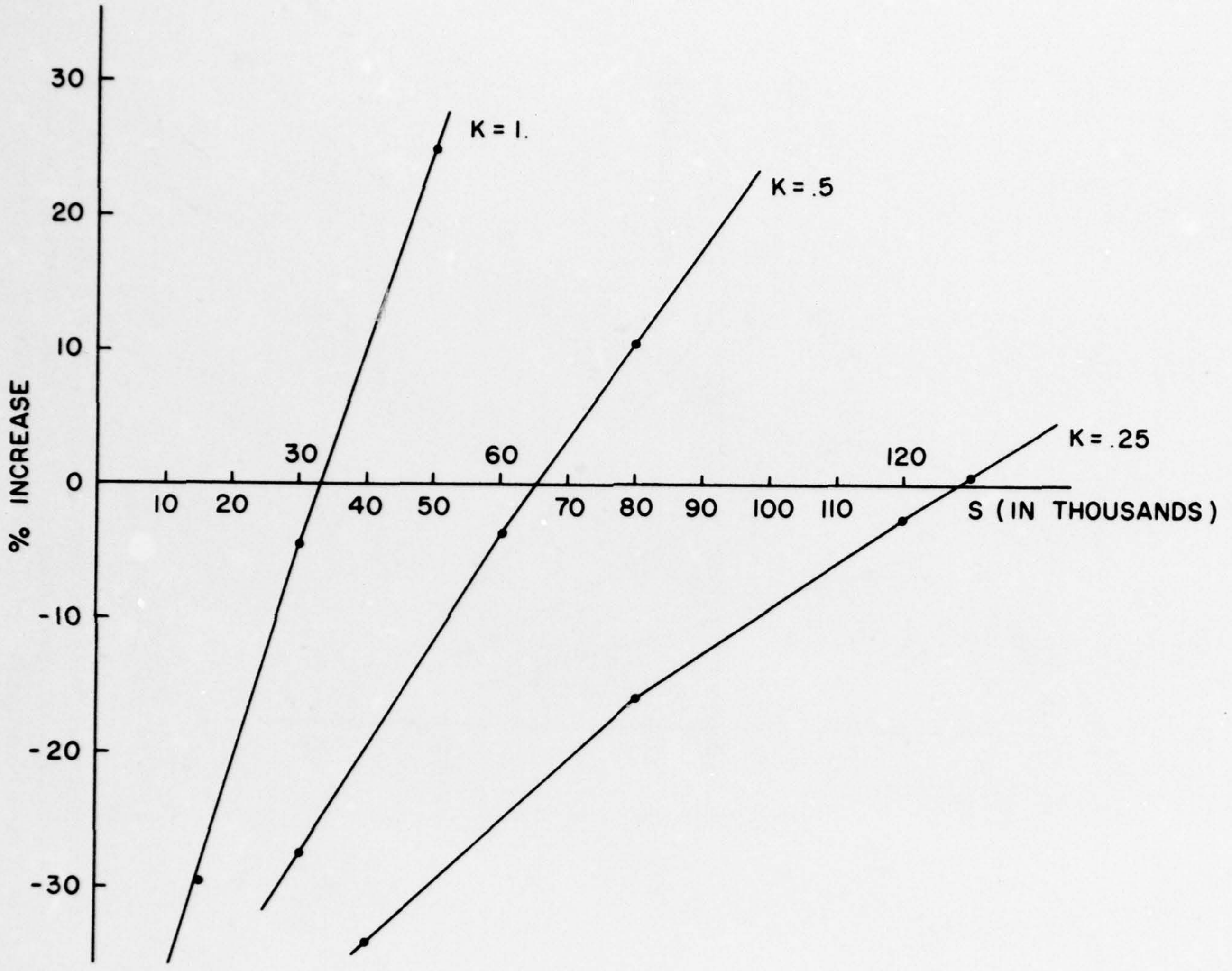


Figure 6

Same as figure 5, except that remote storage costs are only 10 percent of local costs.

just preceding for analysis of these costs.) Most interesting are the facts that network software costs (NCP, etc.) are less than 1% of the total cost, and that (outside of packet charges) the major proportion of network-related costs arise from the delays incurred and from connection setup overhead. Since setup costs are relatively constant, as other factors become larger due to (for example) moving more data or more remote processing the significance of these terms dwindles and more complex protocol negotiations become viable. Figure 7 gives some indication of how protocol costs vary as a function of the number of protocol messages exchanged before the transfer commences. (The amount of data transferred is held constant at 5×10^3 bytes.) A more detailed analysis of the aspects of network overhead is needed, especially with regard to its implications for front-ends. An analysis of the overall impact of network software on the host system would be useful to determine under what circumstances front-ending is a useful tactic.

We also found that increasing network bandwidth had little effect on lowering total cost. For example, with $Z = .1$ and $S = 10,000$ bytes, increasing network bandwidth from 5×10^3 bytes/sec to 5×10^5 bytes/sec resulted in just over a 2% decrease in cost. This implies that for bulk transfers network delay costs are relatively small. However, this does not imply that increasing bandwidth will not be cost-effective. Many highly interactive network activities and/or global traffic levels may require higher bandwidths.

Local caching of data appears to be a useful method for using a network in a cost-effective manner. With this method, the local system maintains a partial copy of the data set. The contents of this copy are determined by the results of past accesses or in some cases by some knowledge of what will be needed. When the user requests data the

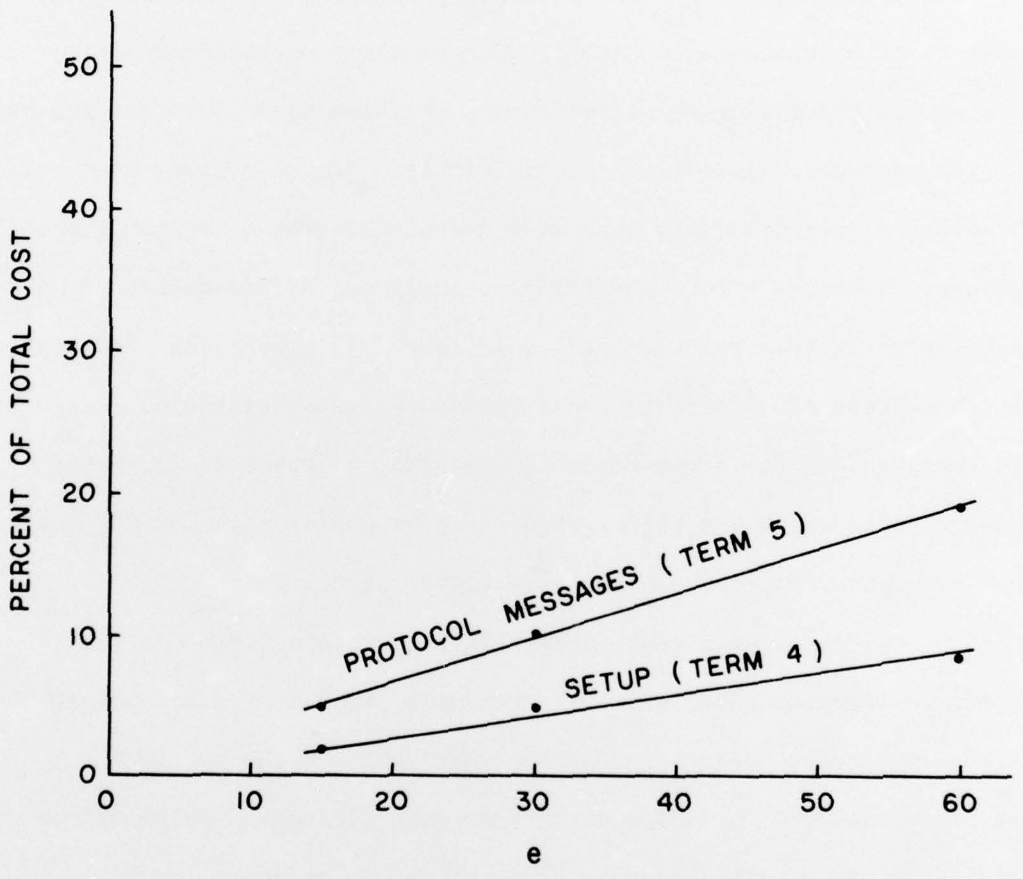


Figure 7

Increase of relative contribution of protocol costs to total cost of remote strategy as e (the number of message exchanges) increases. Only a small data set is assumed transferred, and few accesses are assumed (S = 5000, r = 5000, q = 0).

system first looks to see if the data is local; if so it is fetched from the local storage medium; if not then it must be retrieved from the master copy over the network. This is a sort of "network working set" strategy. Using the model to investigate the properties of such a strategy, we found (see figure 8) a rather steep rise in cost as the fraction of requests that must use the network increased. Of course, whether most requests can be answered locally depends upon the size of the local store, the degree of locality exhibited by the requests and the replacement algorithm used. However, if the fraction of remote requests can be kept low, significant savings can be achieved by the local caching of data. Further work is needed to determine the locality properties of data base activity so that one can determine what the size of the local store must be so that a large fraction of the requests may be satisfied locally.

As we have seen, the major result of this investigation is that heterogeneity must be introduced into a network before remote storage is advantageous for a user, and even then a minimum amount of data should be moved to the remote site and a maximum amount of computing should be done once it's been moved. Interestingly enough, host-related network software overhead does not contribute significantly to the total cost. It is not clear what implications this has for the arguments for front-ending systems; however, a closer look at these problems should be undertaken. An analysis of network software from the point of view of the host operating system rather than from that of a single process is needed to answer the questions generated by these findings.

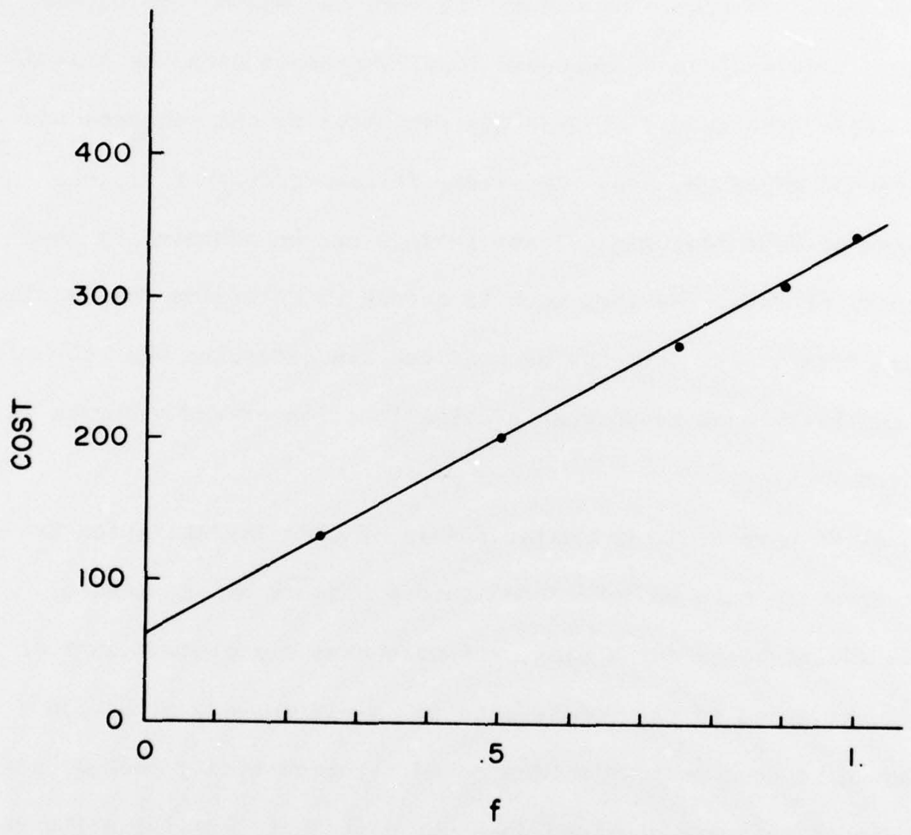


Figure 8

Total cost of responding to a set of requests vs. f, the fraction of the requests requiring remote access.

Plans for Further Work

Clearly, much more can be learned by experimentation with the present model. By using parameters that describe specific systems and their costs, we should be able to develop cost comparisons for important real applications. However, this requires that accurate measurements be made of systems to get useful values for the various parameters. In fact, accurate measurement of the network parameters used in this model are sorely needed, in addition to the refinement of the cost terms to allow the investigation of more complex situations.

We might also investigate other approaches to deciding on a "best" storage policy. For example, since protocol implementations reside as user-level processes in many operating systems, and since it is often useful to consider the data set as being staged in the remote system, it would be interesting to consider an alternative approach which runs as follows: The data set allocations on the remote site are determined according to the LSWL model, and the lowest-cost strategy is selected. The cost of this strategy plus the relevant network costs are then used to form the lowest level of the local hierarchy, where the cost for the local levels is computed using the LSWL model and the last level (the remote one) uses a slightly modified form. Further study is needed to determine whether this approach will yield useful data for decision making.

There are a number of other possible extensions of this study which would be worth pursuing in the future. A few of these extensions, which include both model refinements and useful applications, are listed here.

1. Considering the various terms as independent modules would provide a much more flexible framework in which various system architectures and strategies could be appraised.
2. The effects of the finite size of the storage devices might be included.
3. As mentioned earlier, the definition of the adjusted system cost m does not appear to reflect the effects of increased load on the system. This point requires more investigation to gain a better understanding of this parameter and of how, if necessary, system loads may be inserted into the model.
4. The model developed by Lum et al. was intended to represent file migration or data staging. Thus, when a data set is written back to the inactive device, the operation is considered to be symmetrical to the original read. If this model is to be an accurate characterization of a data management system, it will be necessary to include the cost of performing updates.
5. Since data base reliability appears to be one of the major advantages of distributing, it is very important that the model be capable of evaluating the cost of various multi-copy backup schemes with respect to the level of reliability they provide.
6. It would be worthwhile to consider the arguments for and against front-ending and try to determine under what circumstances front-ending will be advantageous.

References

Casey, R.G.
1972 "Allocation of Copies of a File in an Information Network," Proc. AFIPS Spring Joint Computer Conference, AFIPS Press, Montvale, N.J., pp. 617-625.

Chu, W.W.
1973 "Optimal File Allocation in a Computer Network," Computer-Communications Networks, N. Abramson and F. Kuo, eds., Prentice-Hall, pp. 82-94.

Lum, V.Y.; Senko, M.E.; Wang, C.P.; and Ling, H.
1975 "A Cost Oriented Algorithm for Data Set Allocation in Storage Hierarchies," CACM 18, pp. 318-322.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER
18 CAC Document Number 179 19
JTSA Document Number 5514

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE *(and Subtitle)*
6 Research in Network Data Management and Resource Sharing. A Cost Model for Data Distribution,

5. TYPE OF REPORT & PERIOD COVERED
9 Research Rep[redacted]t,

6. PERFORMING ORG. REPORT NUMBER
CAC-#179

7. AUTHOR(s)
10 John D. Day [redacted]
Geneva G. Belford

8. CONTRACT OR GRANT NUMBER(s)
15 DCA 100-75-C-0021

9. PERFORMING ORGANIZATION NAME AND ADDRESS
2 Center for Advanced Computation
1 ~~University of~~ Illinois at Urbana-Champaign
Urbana, Illinois 61801

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Joint Technical Support Activity
11440 Isaac Newton Square, North
Reston, Virginia 22090

12. ~~REPORT DATE~~
11 1 Nov[redacted], [redacted]975

13. NUMBER OF PAGES
~~45~~ 12 46p.

14. ~~MONITORING AGENCY NAME & ADDRESS~~ *(if different from Controlling Office)*
14 UIUC-CAC-DN-75-179,
CAC-179

15. SECURITY CLASS. *(of this report)*
UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. ~~DISTRIBUTION STATEMENT (of this Report)~~

Copies may be obtained from the
National Technical Information Service
Springfield, Virginia 22151

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

No restriction on distribution.

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

information system modeling
hierarchical storage
network modeling

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

A model is developed to study the costs of storing data at a remote site in a network. The model is basically that of a storage hierarchy in which data is stored at a remote site when inactive and staged to a local, rapid-access device for use.

[illegible]

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

407227

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUC-CAC-DN-75-179 ✓	2.	3. Recipient's Accession No.

4. Title and Subtitle
Research in Network Data Management and Resource Sharing - A Cost Model for Data Distribution

5. Report Date
November 1, 1975

6.

7. Author(s)
John D. Day and Geneva G. Belford

8. Performing Organization Rept. No. CAC #179

9. Performing Organization Name and Address
Center for Advanced Computation
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

10. Project/Task/Work Unit No.

11. Contract/Grant No.
DCA100-75-C-0021

12. Sponsoring Organization Name and Address
Joint Technical Support Activity
11440 Isaac Newton Square, North
Reston, Virginia 22090

13. Type of Report & Period Covered
Research

14.

15. Supplementary Notes

16. Abstracts

A model is developed to study the costs of storing data at a remote site in a network. The model is basically that of a storage hierarchy in which data is stored at a remote site when inactive and staged to a local, rapid-access device for use.

17. Key Words and Document Analysis. 17a. Descriptors
information system modeling
hierarchical storage
network modeling

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement No restriction on distribution Available from the National Technical Information Service, Springfield, VA 22151	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 45
	20. Security Class (This Page UNCLASSIFIED	22. Price